Comment introduire simplement et uniformément deux modes d'interaction asynchrones complémentaires dans un système d'analyse de documents existant

Joseph Chazalon, Bertrand B. Couasnon, Aurélie Lemaitre

Comment introduire simplement et uniformément deux modes d'interaction asynchrones complémentaires dans un système d'analyse de documents existant

J. Chazalon[*] — B. Coüasnon[**] — A. Lemaitre[***]

Université Européenne de Bretagne
[*] Univ. de Rennes 1 — [**] INSA Rennes — [***] Univ. de Rennes 2
IRISA (UMR 6074), Rennes, France
{joseph.chazalon,bertrand.couasnon,aurelie.lemaitre}@irisa.fr

RÉSUMÉ. *L'analyse et la reconnaissance de documents dégradés, en particulier les documents anciens, est difficile. Les systèmes existants recourent généralement à une correction manuelle des résultats en post-processing, sans tirer profit de ces nouvelles informations pour améliorer leur réponse. Cet article explique comment adapter un système d'analyse de documents existant pour lui permettre d'interagir avec un opérateur humain ou d'autres processus, durant la phase d'analyse, et exploiter à ce moment les informations externes fournies. Nous décrivons l'architecture minimale requise, puis les deux modes d'interaction complémentaires que nous proposons : ils sont adaptés au traitement de documents en grand volume, et simples à implémenter. Pour des tâches de transcription de mots manuscrits dans des documents du XVIII^{e} siècle, notre prototype montre une réduction conséquente de la quantité de travail manuel nécessaire.*

ABSTRACT. *Extracting contents from degraded documents, like historical ones, is difficult. Existing analysis systems usually rely on a manual correction of results during a post-processing stage, and cannot make use of external information to adapt their response. This paper presents how to adapt an existing document analysis system to enable an efficient interaction during the analysis stage, and benefit from external information. We describe the minimal architecture required, and the two complimentary interaction models we propose: they are suitable for mass document processing, and easy to implement. For transcription tasks in documents dating from the 18th century, our prototype permitted an important reduction of human workload.*

MOTS-CLÉS : *interaction homme-machine ; interaction asynchrone ; modèles d'interaction ; documents dégradés ; documents anciens*

KEYWORDS: *human interaction; asynchronous interaction; interaction models; degraded documents; historical documents*

1. Introduction

Les difficultés rencontrées lors de l'extraction et l'indexation de contenus de documents dégradés, anciens en particulier, ont deux causes principales. Premièrement, la dégradation du média entraîne une altération du contenu. Les documents sont alors semblables à des canaux de communication bruités qui nécessitent des méthodes d'analyse permettant la correction d'erreurs. Deuxièmement, le contenu et la structure des pages peuvent varier considérablement dans les collections historiques : les contenus imprimés et manuscrits peuvent être mêlés, le scripteur peut changer, le vocabulaire utilisé peut être obsolète ou inconnu à l'avance (patronymes par exemple), les structures ne sont pas contraintes par des systèmes de mise en forme modernes… Les modèles que nous utilisons ont alors tendance à être peu fiables car des cas imprévus seront inévitablement rencontrés au cours de l'analyse des importantes quantités de pages contenues dans les collections historiques que nous traitons.

Ces causes que nous avons identifiées sont à l'origine de nombreuses erreurs au cours du traitement de documents en grand volume. De plus, l'interdépendance contextuelle entre les éléments que nous extrayons démultiplie l'impact de ces erreurs et, par conséquent, la quantité de corrections manuelles nécessaires *in fine*. Dans cette situation, il est souvent plus efficace de demander à un opérateur humain de produire directement les résultats « à la main », plutôt que de devoir corriger en post-traitement toutes les erreurs générées.

Par conséquent, un système d'analyse de documents devrait tirer profit *pendant l'analyse* de l'information externe qui lui est fournie, afin d'améliorer sa réponse pour les pages *passées et futures*, dans l'objectif de *réduire globalement* la quantité de travail humain nécessaire. Bien sûr, cette interaction avec un opérateur humain doit être asynchrone, pour éviter de bloquer l'humain ou le système automatique.

Cet article présente comment deux modes d'interaction complémentaires peuvent être facilement intégrés à un système d'analyse existant en se basant sur un même socle commun. Tout d'abord, en section 2, nous analysons quelques approches interactives existantes pour l'analyse de documents, et montrons que deux modes d'interaction sont nécessaires. En section 3, nous présentons l'architecture minimale requise pour mettre en œuvre ces modes d'interaction. En section 4, nous détaillons comment nous avons transformé notre analyseur de page, grâce à une base commune facilement adaptable à d'autres systèmes, pour permettre l'utilisation de ces deux modes d'interaction. Finalement, en section 5, nous illustrons les capacités d'un tel système sur quelques expériences que nous avons menées.

2. État de l'art

Peu de systèmes d'analyse de documents permettent une interaction avec un opérateur humain, que ce soit pour guider ou corriger la réponse système, au cours de la phase d'analyse. Nous nous intéressons donc ici à quelques-uns de ces systèmes qui illustrent bien les défis auxquels nous devons faire face.

Le système Edelweiss (Roussel *et al.,* 2001), tout d'abord, propose une première forme d'interaction. Lors de l'analyse de pages de documents, il produit progressivement une structure qu'un opérateur humain peut éditer à certains moments au cours de ce processus. Ceci permet d'influencer la réponse du système pour le reste de l'analyse de la page courante, en transformant virtuellement n'importe quel élément dans la structure du document. Cependant, ce système ne remet pas en cause les résultats intermédiaires précédents, et ne valide pas le fait que l'information fournie par un opérateur humain est bien conforme au modèle du document qui n'est pas explicite.

Le système smartFIX (Klein *et al.,* 2004), par ailleurs, illustre bien le besoin d'interagir avec un opérateur humain lors du traitement de données en grand volume, tâche pour laquelle ce système est conçu. smartFIX permet de traiter des documents administratifs, comme des factures par exemple, et de déclencher automatiquement des transactions une fois les données utiles reconnues. Ce système commence par extraire et reconnaitre les éléments utiles, en leur affectant un score de confiance. Pour les éléments problématiques, une correction manuelle est sollicitée, tandis que pour les éléments suspects, une validation humaine est requise. Cette approche permet de montrer la nécessité de permettre une certaine gradation, à la fois dans la quantité d'information demandée à un opérateur humain et dans la façon de guider ou contraindre les réponses de l'opérateur humain. Toutefois, smartFIX limite la possibilité de modifications faites par l'humain aux seuls résultats finaux, et ne permet donc pas de corriger les erreurs originales, ni de guider l'analyse à certains moments clés.

Nagy et Veeramachaneni (Nagy *et al.,* 2008) se sont intéressés aux caractéristiques d'une interaction efficace entre un système de classification et un opérateur humain. En particulier, ils suggèrent que *« l'opérateur ne devrait même pas avoir à regarder les données qui ont été classées sans problème »*, et également que *« chaque interaction devrait être utilisée par le système pour améliorer les tâches de classification subséquentes »*. Leur premier conseil dépend étroitement de ce qu'ils appellent l'interaction « initiée par la machine », où l'opérateur humain réagit aux problèmes qui sont détectés. Cependant, il faut noter que la détection d'erreurs et la localisation de leur cause ne peuvent pas toujours être faites automatiquement, malgré le fait qu'on souhaiterait quand même pouvoir, dans ces cas, remplacer certaines parties du traitement automatique par un traitement manuel.

La capacité à détecter les erreurs est le critère essentiel qui permet de décider quel mode d'interaction il est possible d'utiliser. Nous proposons donc de distinguer deux modes d'interaction. Le premier, celui à privilégier ; est le mode *dirigé* (par la détection automatique d'erreurs) (Chazalon *et al.,* 2011), pour lequel, lorsque la détection d'erreur est possible, le système posera une question (à laquelle un opérateur humain devra répondre) à chaque fois qu'il détectera une erreur. Le second, qui complète le premier, est le mode *spontané* (Chazalon *et al.,* 2012), adapté aux situations dans lesquelles la détection automatique d'erreurs n'est pas possible. Dans ce deuxième mode, l'interaction peut être qualifiée « d'initiée par l'humain » car le système aura à réagir à l'information externe qui lui est fournie, en particulier pour améliorer sa réponse pour une page donnée.

Les sections suivantes présentent donc comment permettre à un système d'adapter sa réponse grâce à une analyse *itérative* de pages que nous proposons, et comment une base commune autorise la mise en place de ces deux modes d'interaction que nous proposons. Précisons que l'amélioration du système grâce à des techniques d'apprentissage artificiel ne sera pas abordé ici, même si l'architecture proposée le supporte en théorie.

3. Architecture pour l'analyse itérative de pages et l'interaction avec un opérateur humain

Pour mettre en place une interaction entre un opérateur humain et un système d'analyse de document qui ait lieu au cours de la phase d'analyse, et qui soit asynchrone (pour éviter que l'opérateur humain ou le système doivent s'attendre mutuellement), tout en permettant la remise en cause de résultats précédents pour les pages traitées, nous avons besoin de :

– un mécanisme d'analyse itérative de pages ;

– une structure de stockage et d'échange que nous appelons *« mémoire visuelle »* ;

– une architecture minimale organisée autour de quelques composants essentiels.

Cette section détaille ces éléments et montre l'importance du composant chargé de l'analyse des pages dans l'exploitation d'informations externes.

3.1. *Vue générale du mécanisme d'analyse itérative*

Le concept d'analyse itérative fait référence à un mécanisme très général. Sa caractéristique essentielle est de reposer sur un système qui supporte l'apport progressif d'informations à propos des images traitées et tire profit de cette connaissance externe pour améliorer sa réponse pour les images concernées.

Plusieurs scénarios d'analyse sont alors possibles. L'un des plus simples qu'on puisse imaginer est le suivant, qui est également celui utilisé pour l'expérience décrite en section 5.2. Dans ce scénario, nous traitons d'abord toutes les pages d'une collection automatiquement, puis nous laissons l'opérateur humain apporter, de façon asynchrone, des informations complémentaires à propos des pages. Chaque page est ensuite traitée à nouveau en tenant compte de ces nouvelles informations. Cette étape d'analyse forme dans le mécanisme global une itération qui sera répétée jusqu'à atteindre un critère d'arrêt : qualité suffisante, coût manuel maximal, ou autre.

Cette approche repose sur la définition d'un objectif commun entre le système automatique et l'opérateur humain, ainsi leur capacité d'échanger entre eux des informations à propos du contenu des pages, et ce selon un protocole pré-établi. L'interaction étant nécessairement asynchrone, le système doit être capable de stocker l'information associées aux différentes pages, et également de traiter les pages indépendamment de l'opérateur humain. L'opérateur humain, quant à lui, fournit autant de réponses que

nécessaire (dans le cas du mode *dirigé*), et doit parfois fournir des éléments supplémentaires (dans le cas du mode *spontané*).

3.2. *Mémoire visuelle*

La *« mémoire visuelle »* est une structure de données persistante associée à chaque image. Elle est utilisée comme support pour l'échange d'information à propos des pages entre processus, et elle est également présente au cœur de l'analyse de page pour permettre de réintégrer et exploiter l'information qu'elle contient.

Chaque processus peut transformer le contenu de cette structure qui associe à chaque donnée contenue une forme et une position dans le référentiel de l'image associée. Afin de permettre, au cours de l'analyse de page, la fusion entre les informations extraites de l'image et celles provenant de « l'extérieur », cette structure doit être rendue disponible en même temps que l'image lors de l'analyse de la page. Il faut par ailleurs garantir que l'information externe est disponible à chaque moment de l'analyse, et autoriser la création, la suppression ou la modification de données contenues dans la mémoire visuelle.

À titre d'exemple, le contenu de la mémoire visuelle associée à une page contenant un tableau de nombres pourrait être composé d'objets géométriques portant des informations sur le type des colonnes, sur la position des nombres et leurs valeurs, sur la position des éléments suspectés d'être mal reconnus, etc. Nous notons $M_I(t)$ la mémoire visuelle associée à une image I à l'instant t.

3.3. *Composants nécessaires*

L'analyse itérative de pages est rendu possible par les composants suivants, comme illustré en figure 1.

3.3.1. *Module de pilotage*

Ce composant gère l'invocation des différents processus lors de l'analyse et leur communication. Il matérialise un niveau d'abstraction où le scénario de l'analyse peut être joué, et où les propriétés liées à la collection peuvent être exploitées.

3.3.2. *Base de données centrale*

La base de données centrale stocke les mémoires visuelles associées à chaque page de la collection. Elle contient donc une connaissance globale à propos de toutes les pages, et cette connaissance croît progressivement au cours de l'analyse.

3.3.3. *Analyseur de pages*

Grâce à un modèle de document spécifique, l'analyseur de pages tente de localiser et de reconnaître les contenus pertinents d'une image I en profitant des informations

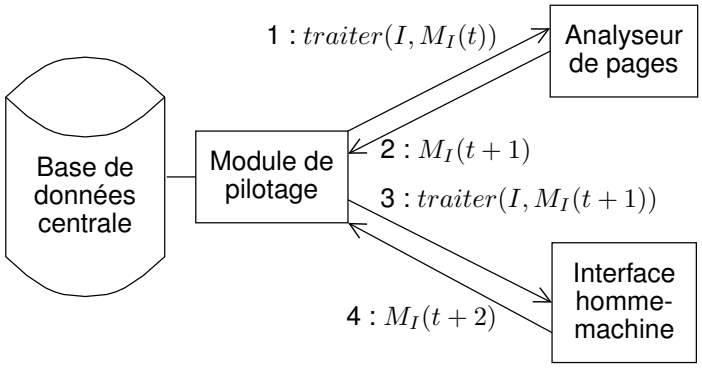

Figure 1. *Une étape simple (qui sera répétée autant que nécessaire au cours de* l'analyse itérative*) d'un échange d'information possible pour une image donnée entre un analyseur de page et une interface homme-machine. Les composants utiles de l'architecture sont visibles, ainsi que la* mémoire visuelle M_I *associée à l'image* I*. À l'instant* t*, le module de pilotage commence par invoquer l'analyseur de page sur l'image* I*, avec la mémoire* $M_I(t)$*. Le module de pilotage récupère* $M_I(t+1)$ *en résultat de l'analyse, et transfert cette mémoire à l'interface homme-machine, qui, à son tour, produit* $M_I(t+2)$ *et la renvoie au module de pilotage. En cas réel, les couples d'opérations (1, 2) et (3, 4) sont effectués indépendants pour des images distinctes, ce qui permet une communication asynchrone entre l'analyseur de pages et l'interface homme-machine.*

contenues dans la mémoire visuelle associée $M_I(t)$. Le mécanisme d'analyse itérative l'autorise à transformer $M_I(t)$ et renvoyer au module de pilotage, une fois le traitement terminé, le contenu mis à jour $M_I(t+1)$. À chaque itération, l'analyseur de page *réanalyse toute l'image* (certaines optimisations non détaillées ici permettent d'alléger les calculs) et tire profit de l'information externe fournie de façon asynchrone. Cela permet de produire de nouveaux résultats structurés, valides au regard du modèle de page. Dans le cas d'une interaction en mode dirigé, l'analyseur de page peut générer des questions pour indiquer qu'il manque de connaissances pour pouvoir gérer une erreur détectée.

3.3.4. *Interface homme-machine*

Lorsqu'elle est sollicitée par le module de pilotage, l'interface homme-machine autorise l'opérateur humain à changer le contenu de la mémoire visuelle de certaines images, au même titre que les autres processus du système. Il est en effet important que l'analyseur de page et l'interface homme-machine soient capables de produire des *éléments similaires* dans la mémoire visuelle associée à chaque page, et ce afin de leur permettre de collaborer d'une telle sorte que l'opérateur humain puisse ponctuellement *prendre la place de certaines parties de l'analyse automatique et produire des résultats intermédiaires*. Nous considérons l'interface homme-machine d'un point de

vue théorique dans cet article car nous nous focalisons sur l'implémentation de l'analyseur de page.

4. Implémentation uniforme pour les modes d'interaction spontané et dirigé

Nous avons vu que l'analyseur de page joue un rôle critique dans la mise en place d'une interaction. Dans des articles précédents, nous avons séparément montré l'intérêt de l'interaction qu'elle soit selon un mode *dirigé* (voir (Chazalon *et al.,* 2011, Guichard *et al.,* 2011), et la section 5.1) ou un mode *spontané* (voir (Chazalon *et al.,* 2012) et la section 5.2). Ces deux modes permettent, lorsqu'ils sont utilisés à bon escient, de réduire la quantité de corrections manuelles nécessaires, pour un niveau de qualité donné, par rapport à un fonctionnement basé sur des corrections exclusivement en post-traitement.

Cette section montre comment nous avons pu intégrer ces deux modes d'interaction au sein d'un même analyseur de page, en identifiant une base commune permettant une mise en œuvre homogène. Il est alors possible de gérer simultanément des cas où la détection d'erreur automatique est possible, et des cas où elle ne l'est pas. Nous commençons par décrire rapidement le système DMOS-P pour lequel le prototype d'implémentation a été développé, puis nous détaillons chaque mode d'interaction et montrons comment ils peuvent être mise en œuvre grâce à une base commune. Précisons que notre approche devrait pouvoir être appliquée à tout autre système d'analyse de documents guidé par un modèle de page, pourvu qu'il soit possible d'y intégrer un mécanisme de mémoire visuelle.

4.1. *Framework DMOS-P et langage de description associé*

DMOS-P (Coüasnon, 2004) est un framework générique d'analyse de documents qui permet la génération automatique d'analyseurs dédiés à un type de document grâce à des descriptions grammaticales spécifiques. La description des pages est réalisée à l'aide d'une extension à 2 dimensions des *Definite Clause Grammars*, appelée *Enhanced Position Formalism* (EPF). L'exemple suivant illustre brièvement la syntaxe de ce langage : pour reconnaître `A`, nous essayons, en `haut` de l'image, de reconnaître `B`, et, si ce n'est pas possible, nous essayons en `bas` de reconnaître `C`. Les dérivations sont notées « `::=` », la concaténation est notée « `&&` » et les alternatives sont représentées par plusieurs dérivations pour le même membre gauche d'une règle.

```
A ::= AT(haut) && B. % clause 1, essayée en premier
A ::= AT(bas) && C. % clause 2, essayée si 1 échoue
```

L'opérateur « `AT` » est utilisé pour le positionnement de la « tête de lecture » de l'analyseur : il définit une zone dans le plan de l'image, et seuls les éléments présents dans cette zone seront considérés dans la règle qui suit cet opérateur.

À l'instar des grammaires attribuées, les règles ont des paramètres d'entrées (hérités) et de sortie (synthétisés) indiqués par des signes « + » et « - » comme indiqué :

```
reconnaitreNombre(+PositionNb, -Valeur) ::=
   appelClassifieur(+NumPos, -Valeur).
```

4.2. *Extensions de DMOS-P pour l'interaction dirigée*

Nous avons récemment introduit le concept d'interaction en mode dirigé (Chazalon *et al.,* 2011) et montré son intérêt et son efficacité avec quelques exemples (Guichard *et al.,* 2011). Ce mode d'interaction repose sur l'introduction de 3 nouveaux opérateurs dans le langage de description de page *EPF*. La sémantique associée à ces opérateurs permet d'exprimer des propriétés utiles dans le modèle de page. Leur implémentation permet de modifier le comportement de l'analyseur de pages pour gérer automatiquement l'interaction entre l'analyseur de pages et un opérateur humain, ou d'autres composants du système. Cette implémentation est basée sur un échange asynchrone d'informations fait de questions et de réponses en utilisant la mémoire visuelle, et ce au cours des différentes invocations de l'analyseur de page, à chaque pas de l'analyse itérative.

La mémoire visuelle est alors le vecteur de ces *questions* (notées Q) et *réponses* (notées R) qu'elle permet de localiser dans le référentiel de l'image. Les containers suivants sont utilisés :

$Q(\mathit{Texte}, \mathit{TypeDonnée})$
$R(\mathit{Donnée})$

Texte est généré par le système et décrit le problème rencontré, et *TypeDonnée* indique le type du contenu attendu pour la réponse. Le contenu de la réponse est stocké dans l'attribut *Donnée* de la réponse.

D'un point de vue technique, ces opérateurs sont implémentés grâce au mécanisme de *continuations* disponible dans le langage de programmation logique utilisé par DMOS-P. Les continuations peuvent être assimilées, dans notre cas, à des constructions similaires au mécanisme d'exceptions bien connu et disponible dans la plupart des langages. Par conséquent, notre approche peut facilement être transposée à d'autres systèmes existants.

L'implémentation que nous proposons permet la mise en place d'un mécanisme de détection et de correction d'erreurs de façon asynchrone. Elle permet également une récupération sur erreur lors de l'analyse d'une page. Afin de rendre leur compréhension et leur utilisation plus facile, nous avons choisi de faire une analogie forte avec le mécanisme d'exception dans les noms qui sont données aux opérateurs, comme montré juste après. Ceci permet de mieux anticiper leur comportement lors de l'analyse. Ces opérateurs, que nous allons à présent détailler, sont conçus pour :

– détecter des erreurs et poser des questions (l'opérateur associé contient le mot-clé *« raise »* en référence à la gestion d'exceptions) ;

– garantir que les réponses sont utilisées par l'analyseur de pages pour progresser, et éviter qu'il entre dans une boucle infinie et repose les mêmes questions sans arrêt (l'opérateur associé identifie une partie de la description pouvant contenir une erreur et contient alors le mot-clé *« try »*) ;

– progresser autant que possible dans l'analyse des parties indépendantes d'une page, même si un problème se produit (l'opérateur associé permet de reprendre l'analyse et contient alors le mot-clé *« catch »*).

4.2.1. *Poser des questions (Détection d'erreurs)*

L'opérateur de détection d'erreur permet d'indiquer qu'une partie de la page n'a pas pu être interprétée correctement. Ceci peut avoir plusieurs causes :

– aucune interprétation valide n'a pu être proposée (toutes les alternatives possibles pour un élément requis ont échoué) ;

– les éléments reconnus sont incohérents entre eux (non respect de l'incrément dans une liste de nombres par exemple) ;

– le cas courant serait mieux traité par un opérateur humain : méthode non implémentée, peu fiable, etc.

La syntaxe et l'algorithme simplifié décrivant le comportement de l'analyseur de page sont les suivants :

```
raiseQuestion(+Texte, +Zone, +TypeDonnée)
```

Lorsque l'analyseur entre dans une partie de la grammaire annotée avec cet opérateur, il réalise les opérations suivantes : 1) il ajoute une question dans la mémoire visuelle M_I, avec une forme et une position définies par `+Zone` qui localise le problème ; et 2) il continue l'analyse juste après la dernière invocation de `catchQuestion`.

L'exemple suivant illustre comment poser une question si, dans un courrier sur nous essayerions de reconnaître, l'analyseur n'arrive à localiser le bloc relatif au destinataire.

```
courrier ::= AT(haut_droit) && destinataire(-Dest)
   && % utiliser l'information relative au destinataire
courrier ::= raiseQuestion("Où est le bloc du
      destinataire ?", page_complete, destinataire).
```

La réponse aux questions est faite hors de l'analyseur de page, grâce à l'interface homme-machine, après que l'étape d'analyse automatique dans l'itération courante soit terminée. Les réponses sont stockées dans la mémoire visuelle et seront disponibles lors de la prochaine itération.

4.2.2. *Utiliser les réponses (Correction d'erreurs)*

L'opérateur de correction d'erreur sert à indiquer qu'une partie de la description de la page (modèle grammatical) devra être ignorée si une réponse (information externe) existe dans la mémoire visuelle associée à l'image courante. Lorsque l'analyseur arrive dans un cas correspondant à cette partie de la description de la page (par exemple,

pour reconnaitre un nombre), il est possible qu'une erreur survienne et empêche l'analyseur de produire un résultat intermédiaire. Dans ce cas, l'analyse automatique correspondant à la partie de la description annotée avec l'opérateur (au sens syntaxique) de correction sera remplacée par un opérateur humain, ou un autre processus. L'opérateur de correction rend possible *les 2 modes d'interaction*, comme nous le verrons en section 4.3. Sa syntaxe et l'algorithme décrivant le comportement de l'analyseur sont les suivants :

```
getAnswerOrTry(+TypeDonnee, -Resultat, +Regle)
```

où `+TypeDonnee` indique le type du contenu des réponses acceptables, et `+Regle` est une règle qui possède un unique paramètre de sortie dont le type est également `+TypeDonnee`. Le type de l'élément contenu dans la réponse est le même que celui du résultat produit par la règle automatique, ce qui permet de contraindre dynamiquement le domaine des réponses que l'opérateur humain peut fournir.

Lorsque l'analyseur entre dans une partie de la grammaire annotée avec cet opérateur, il vérifie s'il existe une réponse $R(Donnee)$ dans M_I dans la zone de recherche courante pour laquelle le type de l'attribut $Donnee$ est `+TypeDonnee`. Si une telle réponse existe, la valeur de `-Resultat` sera $Donnee$. Sinon, l'analyseur invoque la règle `Regle` et la valeur de `-Resultat` sera celle du paramètre de sortie de `Regle`.

Afin d'éviter que l'analyseur reste bloqué sur la même question indéfiniment, nous avons besoin de garantir que si une réponse appropriée est fournie, la question originale ne serait plus posée à nouveau. Pour ce faire, nous contrôlons que l'utilisation de `raiseQuestion` est faite exclusivement à l'intérieur de parties du modèle annotées avec un opérateur `getAnswerOrTry` avec le même `TypeDonnee` (`NOMBRE` ici). L'exemple suivant illustre cette utilisation.

```
reconnaitreNombre(+PositionNb, -Valeur) ::=
   getAnswerOrTry(NOMBRE, -Valeur,
      appelClassifieur(+PositionNb, -Valeur))

appelClassifieur(+PositionNb, -Valeur) ::=
   % appel sous-processus
   call_lib("classifiers.so", NOMBRE,
            +PositionNb, -Valeur, -Score) &&
   % échec si le score est trop faible
   Score > SEUIL.

appelClassifieur(+PositionNb, -Valeur) ::=
   raiseQuestion("Quel est ce nombre ?",
            +PositionNb, NOMBRE).
```

4.2.3. *Continuer l'analyse (Récupération sur erreur)*

L'opérateur de récupération sur erreur indique la portée maximale d'une erreur dans le modèle de page. Hors de cette portée, l'analyse est suffisamment peu impactée par l'erreur et peut donc être continuée malgré le problème détecté. Cet opérateur est noté :

```
catchQuestion(+Regle)
```

où `+Rule` est une règle quelconque. Lorsque l'analyseur entre dans une partie de la grammaire annotée avec cet opérateur, il invoque la règle `Regle` et intercepte toutes demandes d'interaction lancées avec l'opérateur `raiseQuestion` dans cette règle. Pour une tâche de reconnaissance de table où il serait possible de traiter les différentes lignes indépendamment, voici un exemple d'utilisation possible de cet opérateur.

```
listeLignes ::= catchQuestion(une_ligne) &&
   % detecter les autres lignes indépendamment
```

4.3. *Introduction de l'interaction spontanée*

L'interaction spontanée se base sur un protocole plus simple (moins contraint) que l'interaction dirigée. Par ailleurs, il n'est pas possible, par hypothèse, de détecter les erreurs et donc de poser des questions, et de permettre une récupération sur erreur. Toutefois, la capacité de l'analyseur de page à tolérer qu'une partie de l'analyse automatique soit remplacée par une correction manuelle faite par un opérateur humain reste intéressante. L'opérateur `getAnswerOrTry` contient toute la logique nécessaire à la mise en œuvre d'une interaction spontanée : si nous autorisons l'opérateur humain (grâce à quelques modifications dans le protocole défini dans le module de pilotage) à fournir des informations avec un type acceptable et correctement positionnées dans le référentiel de l'image, alors l'analyseur exploitera automatiquement ces dernières et ignorera certaines parties de l'analyse automatique. Ainsi, l'opérateur humain ne fournit plus juste des réponses, mais plus généralement des informations utiles à propos du contenu de l'image.

Grâce à l'annotation de certaines parties du modèle de page, on peut indiquer quelles seront les parties de l'analyse qui pourront être remplacées par un opérateur humain au cours des phases d'interaction. Dans ce cas, les données produites par l'opérateur humain permettront de guider ou corriger la réponse de l'analyseur de page sur certaines données. Ceci permet, par exemple, de corriger des éléments erronés ou d'ajouter des éléments manquants dans une structure intermédiaire, comme illustré dans l'exemple suivant :

```
ligne_de_nombres::= AT(debut_ligne) &&
   suite_de_nombres.

suite_de_nombres ::=
   getAnswerOrTry(nombre, -Nb,
      trouver1Nombre(-Nb)) &&
   AT(droite_de(+Num)) &&
   % répéter tant que des nombres sont trouvés
```

Si le système rate un nombre dans une page, l'opérateur humain peut l'ajouter dans la mémoire visuelle associée à cette page et, lors de la phase d'analyse de la prochaine itération, le nombre sera utilisé comme s'il avait été détecté dans l'image.

4.4. *Avantages de l'implémentation proposée*

Cette implémentation ne permet pas seulement d'introduire les deux modes d'interaction que nous proposons dans le même analyseur de page, elle a aussi deux autres avantages. Le premier est de permettre une séparation claire entre la description de la page et le comportement de l'analyseur lié à la gestion automatique de l'interaction avec un opérateur humain (ou un processus automatique). Cela permet de garder des modèles de page simples, et aussi d'autoriser l'amélioration de la gestion de l'interaction sans changer le modèle de page.

L'autre avantage est de réintégrer l'information externe (fournie par un humain ou un processus automatique) exactement comme si elle avait été extraite de l'image. Par conséquent, nous pouvons facilement affecter un score de confiance aux informations externes, vérifier qu'elles sont conformes au modèle de page, ou encore les utiliser pour produire les résultats finaux.

5. Applications et résultats

En dehors d'une utilisation pour la production de données à destination d'archivistes, nous avons testé séparément l'intérêt de chaque mode d'interaction proposé.

5.1. *Interaction dirigée pour la transcription de documents du XVIII^{e} siècle*

L'interaction en mode dirigé a été utilisée avec succès pour transcrire des noms manuscrits contenus dans des registres de vente datant du XVIII^{e} siècle (Guichard *et al.,* 2011). La figure 2 montre un exemple de page traitée, et la figure 3 montre quelques noms extraits. Dans les tests que nous avons menés, 70 documents (1206 instances de mots) ont été reconnus selon le processus suivant. Après l'étape de localisation par un analyseur de page, les mots extraits dans plusieurs pages sont groupés par similarité visuelle en clusters (tirant ainsi profit de la redondance de certains mots au cours des pages dans cette collection), et ces clusters sont ensuite annotés manuellement si nécessaire (indice de confiance faible). L'analyse itérative a permis de réintégrer les informations externes (labels plus fiables grâce au contexte de collection) et de valider leur cohérence au sein d'un modèle de page unique. Cette approche globale nous a permis de diminuer le travail d'annotation manuel de 21% (correction manuelle de tous les éléments suspects) à 15% (gain de 28%) pour un taux de reconnaissance total de 80%.

5.2. *Interaction spontanée pour la correction de problèmes de segmentation de nombres dans des documents du XVIII^{e} siècle*

Nous avons également validé l'intérêt de l'interaction en mode spontané pour corriger les problèmes de segmentation se produisant lors de la localisation de nombres

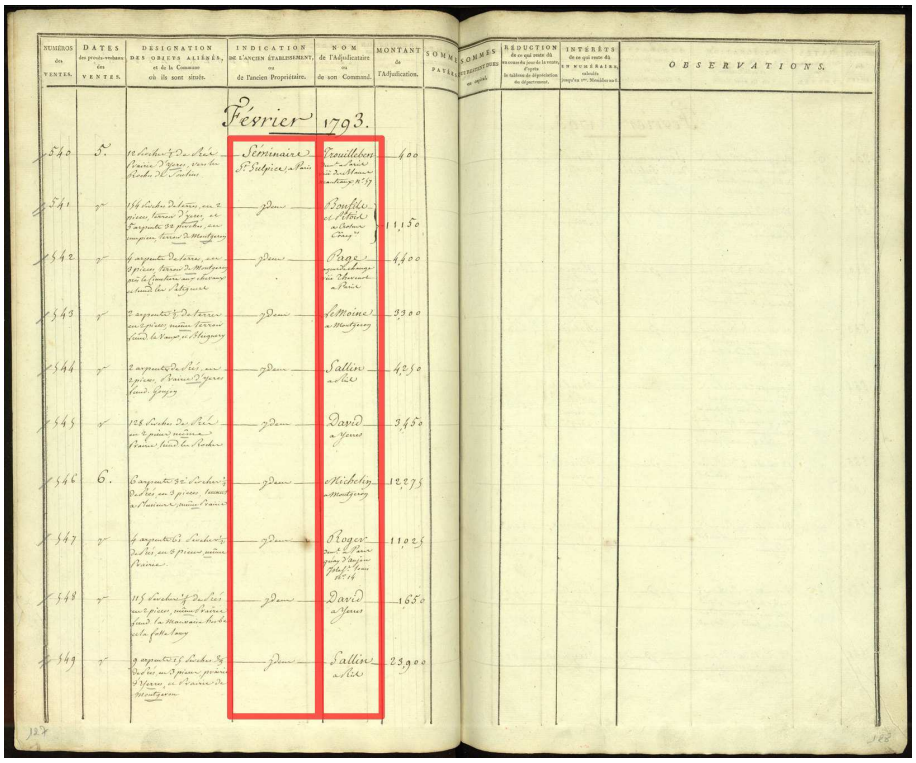

Figure 2. *Extrait de documents du XVIII^e siècle. Les colonnes entourées sont celles contenant les noms à localiser et reconnaître.*

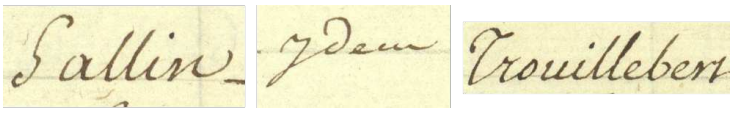

Figure 3. *Exemples de noms extraits :* « SALLIN », « IDEM », *et* « TROUILLEBERT ».

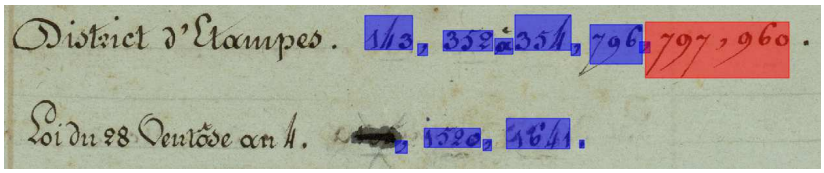

Figure 4. *Extrait de document montrant la détection automatique de séparateurs et de nombres. Un séparateur a été raté à la fin de la première ligne, causant la sous-segmentation des deux derniers nombres.*

dans un extrait d'un document (50 pages avec 1637 vignettes de nombres) datant du XVIII^{e} siècle (Chazalon *et al.,* 2012). La figure 4 montre un extrait de page où les deux derniers nombres de la premières ligne ont été fusionnés par erreur. Nous avons procédé selon le protocole suivant. Ces nombres étaient séparés par des caractères spéciaux qui n'étaient pas toujours détectés correctement, ce qui causait des problèmes de sous-segmentation. Dans les tests que nous avons menés, nous avons permis à un opérateur humain de visualiser les résultats de la localisation, et d'indiquer la position des séparateurs qui n'étaient pas correctement détectés. Cette action d'édition était plus rapide que la localisation manuelle de nombre, qui était alors faite automatiquement par l'analyseur de page. Grâce à cette approche, nous avons pu récupérer 40,6% des vignettes de nombres manquantes et éliminer 30,0% des zones mal détectées. Réintégrer la correction d'erreurs au cours de la phase d'analyse des pages nous permis de diminuer de 29,8% le nombre d'actions manuelles requises pour atteindre ce niveau de qualité (89,2% de vignettes bien localisées) par rapport à une correction en post-traitement.

5.3. *Cas réels et interactions complexes*

Pour traiter des cas réels, il est possible de permettre pour un même modèle de page des interactions variées : spontanées et dirigée, avec différents types de questions et de données. Toutefois, nous n'avons pas encore été en mesure d'évaluer le gain en terme de coût en travail manuel pour des cas réels à grande échelle car il devient très rapidement incohérent de comparer des actions aux coûts différents (exemple : localisation d'une colonne dans un tableau, saisie de la valeur d'un nombre). Il faudrait donc se baser sur une mesure absolue du coût de travail manuel, comme le temps écoulé, pour comparer différentes approches. Cette solution est délicate car elle est extrêmement sensible aux interfaces homme-machine utilisées.

Par ailleurs, on peut signaler que la frontière entre interaction dirigée et interaction spontanée est floue car dans certains cas la détection d'erreur peut être partielle ou imprécise. Dans un exemple basé sur le cas de la section 5.2, où on essaie de détecter et reconnaitre des nombres avec des problèmes de sous-segmentation, on peut parfaitement imaginer écrire une grammaire qui poserait des questions comme celle-ci : *« Je trouve dans cette zone des nombres anormalement longs, manque-t-il des séparateurs ou est-ce correct ? »* à laquelle il serait possible soit de répondre *« La ligne est correctement segmentée. »*, soit d'indiquer des séparateurs manquants (autant que l'opérateur humain le souhaite).

6. Conclusion

Interagir avec un opérateur humain pendant la phase d'analyse est nécessaire pour limiter le coût de traitement dans le cas de documents anciens et dégradés. L'interaction en mode dirigé est certainement le mode le plus efficace dans la majorité des cas, car le fait que le système soit capable de réclamer l'information qui lui manque per-

met de limiter l'interaction avec un opérateur humain au strict nécessaire. Toutefois, la détection automatique d'erreurs n'est pas toujours possible. L'interaction en mode spontané est une alternative efficace dans ces cas, car ce mode permet souvent de réduire la charge de travail manuel par rapport à une correction exclusivement au cours du post-traitement. Cet article a montré comment nous avons introduit ces deux modes d'interaction dans un même analyseur de pages pour permettre de faire face aux deux types de situations rencontrées. L'implémentation associée que nous avons proposée permet de mettre en place un mécanisme de détection, de correction et de récupération sur erreur, de façon asynchrone, au cours des itérations d'analyse faites par le système. Cette implémentation est basée sur des constructions classiques, très proches du mécanisme d'exception des langages de programmation courants, et devrait donc pouvoir être transposé facilement dans un autre système existant. Notre approche permet donc de tirer profit de façon presque transparente d'informations produites hors de l'analyseur de pages lors du traitement de ces dernières. Nous avons ainsi pu exploiter l'information fournie par un opérateur humain, ainsi que du contexte offert par la collection (redondance de mots dans des pages voisines) pour faciliter la transcription de documents datant du XVIII^{e} siècle.

Remerciements

Ce travail a été effectué en collaboration avec les Archives Départementales des Yvelines, en France, avec le support du Conseil Général des Yvelines.

7. Bibliographie

Chazalon J., Coüasnon B., « Iterative analysis of document collections enables efficient human-initiated interaction. », *Document Recognition and Retrieval XIX, Proc. SPIE*, 2012.

Chazalon J., Coüasnon B., Lemaitre A., « Iterative Analysis of Pages in Document Collections for Efficient User Interaction », *Proc. of ICDAR*, 2011.

Coüasnon B., « Dealing with Noise in DMOS, a Generic Method for Structured Document Recognition : An Example on a Complete Grammar », *in* , Lladós, , Kwon (eds), *Graphics Recognition*, vol. 3088 of LNCS, Springer, 2004.

Guichard L., Chazalon J., Coüasnon B., « Exploiting Collection Level for Improving Assisted Handwritten Words Transcription of Historical Documents », *Proc. of ICDAR*, 2011.

Klein B., Dengel A., Fordan A., « smartFIX : An Adaptive System for Document Analysis and Understanding », *in* , Dengel, , Junker, , Weisbecker (eds), *Reading and Learning*, vol. 2956 of LNCS, Springer, 2004.

Nagy G., Veeramachaneni S., « Adaptive and Interactive Approaches to Document Analysis », *in* , S. Marinai, , H. Fujisawa (eds), *Machine Learning in Document Analysis and Recognition*, vol. 90 of *Studies in Computational Intelligence*, Springer, p. 221-257, 2008.

Roussel N., Hitz O., Ingold R., « Web-based cooperative document understanding », *Proc. of ICDAR*, p. 368-373, 2001.